Homework 07 – Starbase Command

Authors: Sumit, Julia, Cindy, Ariel, Maddy, Andrew, Jack

Problem Description

Commander's Log, Stardate 52613.31: All was going well until the Scala Dominion attacked our base. In alliance with the C# Confederacy, they have developed a new weapon that has sabotaged all the graphical user interfaces on this base. If we don't develop new GUIs within the next week, Starbase 1331 will be destroyed.

Solution Description

For this assignment, you will be designing an aesthetically pleasing GUI that meets all the requirements outlined below.

`Starbase.java`

This file contains the `Starbase` class which will be the base for your JavaFX application. This GUI will keep track of what port a starship is in.

- The title of the window should be "Starbase Command"
- The application background should be an image of space. Include this image in your submission as **space.jpg.** *(Hint: One approach is to utilize a StackPane here)*
- The size of your window should be reasonable for the application.
- Your application will consist of 3 main components:
    - A header that says, "Welcome to Starbase 1331!"
    - An area to dock 8 Starships (more on this later)
    - Necessary input fields
- Input Fields:
    - A text input field to enter the name of the incoming Starship
        - Starships can't have empty names or names that are only whitespace
        - It should be clear to the user what the input field is for.
    - A dropdown menu for the Starship type
        - Utilize the enum *Starship.java*
    - A button that docks the incoming Starship at an available port:
        - When the docking button is clicked:
            - Inputs should be checked to make sure they are valid using the following procedures:
                - If the starship can be docked, follow the proper docking procedure (see below)
                - If the starship cannot be docked, an alert message should pop up saying: "[Starship Name] did not receive docking clearance!"

        - All input fields should be cleared after clicking this button
    - A button that is visually different that evacuates the base and clears all docked starships when clicked
- Space Dock:
    - There must be 8 distinct regions which will serve as the docking points:
        - Any port can hold any type of Starship
    - A Starship requesting docking clearance can be placed at any available port. For example:
        - If an INTREPID type ship arrives at an empty base, it will get a port
        - If a GALAXY type ship arrives and all ports are occupied, an alert message should pop up.
    - An occupied port should look visually different from an empty one
        - Occupied ports should display the name and class of the starship
        - Empty ports should say "EMPTY"
        - Occupied ports should be visually distinct from empty ports
    - Clicking on a particular docking port itself should empty it
- **For full credit, your submission must not throw any exceptions to the command line when it is run!**
- **[Optional]** Extra credit will be awarded on the following grounds:
    - Adding a non-trivial addition to the program's graphics that clearly extends its look [e.g. add icons representative of the type of a docked starship]
    - Adding a non-trivial addition to the program's controls or functionality that clearly extends its capabilities [e.g. have a queue for ships that have not gained clearance]
    - An impressive, simply awesome, spectacular submission [e.g. keeping track of docking history in an external file with time stamps]
    - If you choose to go for the extra credit, attach a file **extra_credit.txt** to your submission that explains exactly what additional features you have added.
    - Extra credit features should not affect any of the base functionality!

`Starship.java`

This file contains the `Starship` enum which enumerates all possible Starship types.

Example GUI:

Note: Your submission does not need to match exactly but should meet all the outlined requirements and have proper functionality. This is an open-ended assignment so BE CREATIVE! Much like in professional software engineering, you will need to be able to make design choices on your own for this assignment.

Figure 1. Example application for an empty starbase.

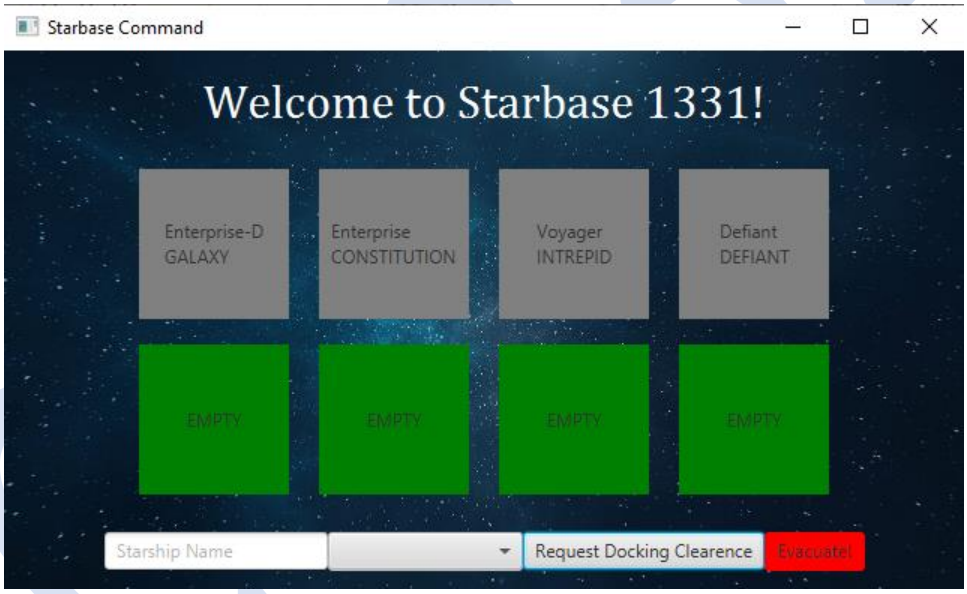

Figure 2. Example application for populated starbase.

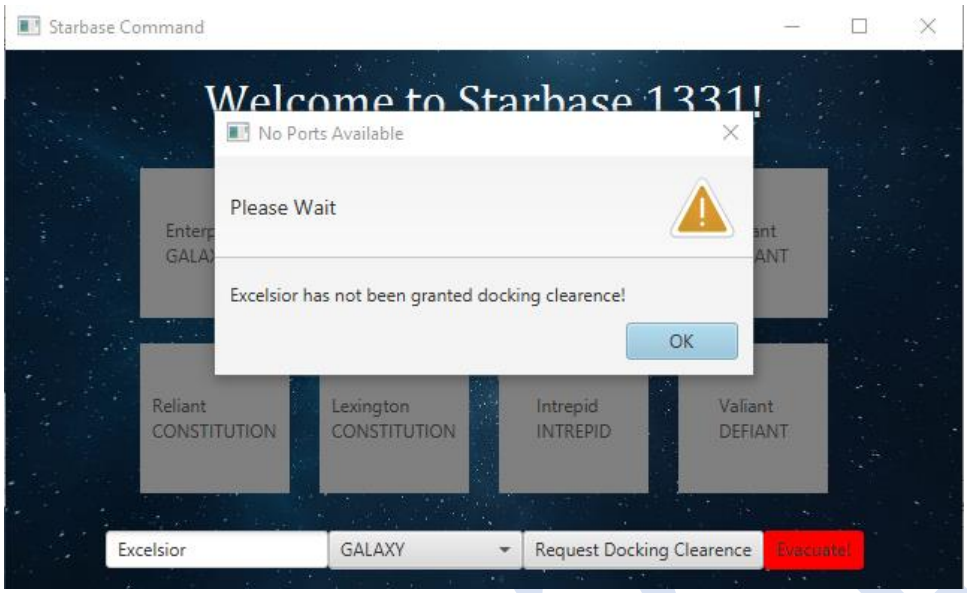

Figure 3. Example for when a Starship was not granted docking clearance.

Tips and Tricks

- The JavaFX API is your friend. Use it to your advantage.
- Make sure you are properly able to run JavaFX programs before starting this assignment. Do NOT wait until the last moment to configure your JavaFX!
- Work incrementally. Get a basic UI up and running. Add the buttons, cells for display, etc., piece by piece. Think of which type of layout manager(s) you want to use.

Rubric

You will be graded on the following items:

- ✓ Window Title
- ✓ Background Image
- ✓ Window Is Reasonably Sized
- ✓ Header Text
- ✓ Input Fields
  - Name Input Field
  - Ship Type Dropdown Menu
  - Docking Button
  - Evacuate Button
- ✓ Docking Ports
  - 8 Ports
  - Port Functionality
  - Port Appearance
  - Docking Logic
- ★ **[Optional] Extra Credit**
  - **Non-Trivial Graphics Addition**

- **Non-Trivial Controls Addition**
- **Subjectively Amazing Submission**
- **Text File Describing Your Additions**

The CheckStyle cap for this homework assignment is 25. Up to 25 points can be lost from CheckStyle errors.

We reserve the right to adjust the rubric, but this is typically only done for correcting mistakes.

Allowed Imports

You can import anything from the *javafx* library and anything from *java.io*

Feature Restrictions

There are a few features and methods in Java that overly simplify the concepts we are trying to teach or break our auto grader. For that reason, do not use any of the following in your final submission:

- `var` (the reserved keyword)
- `System.exit`

Collaboration

Collaboration Statement

To ensure that you acknowledge a collaboration and give credit where credit is due, **we require that you place a collaboration statement as a comment at the top of at least one .java file that you submit**. That collaboration statement should say either:

I worked on the homework assignment alone, using only course materials.

or

In order to help learn course concepts, I worked on the homework with [give the names of the people you worked with], discussed homework topics and issues with [provide names of people], and/or consulted related material that can be found at [cite any other materials not provided as course materials for CS 1331 that assisted your learning].

Allowed Collaboration

When completing homeworks for CS1331 you may talk with other students about:

- What general strategies or algorithms you used to solve problems in the homeworks
- Parts of the homework you are unsure of and need more explanation
- Online resources that helped you find a solution
- Key course concepts and Java language features used in your solution

You may **not** discuss, show, or share by other means the specifics of your code, including screenshots, file sharing, or showing someone else the code on your computer, or use code shared by others.

Examples of approved/disapproved collaboration:

- **approved**: "Hey, I'm really confused on how we are supposed to implement this part of the homework. What strategies/resources did you use to solve it?"
- **disapproved**: "Hey, it's 10:40 on Thursday... Can I see your code? I won't copy it directly I promise"

In addition to the above rules, note that it is not allowed to upload your code to any sort of public repository. This could be considered an Honor Code violation, even if it is after the homework is due.

Turn-In Procedure

Submission

To submit, upload the files listed below to the corresponding assignment on Gradescope:

- **`Starbase.java`**
- **`space.jpg`**
- **`[Optional] extra_credit.txt`**

Make sure you see the message stating "HW07 submitted successfully".

You can submit as many times as you want before the deadline, so feel free to resubmit as you make substantial progress on the homework. We will only grade your last submission, so be sure to **submit every file each time you resubmit**.

Gradescope Autograder

The Gradescope tests serve two main purposes:

- Prevent upload mistakes (e.g. non-compiling code)
- Provide basic formatting and usage validation

Be sure to thoroughly test your code by considering edge cases and writing your own test files. You also should avoid using Gradescope to compile, run, or Checkstyle your code; you can do that locally on your machine.

This assignment will be graded by a TA once the submission deadline has passed, so the output on Gradescope may not necessarily reflect your grade for the assignment.

Important Notes (Don't Skip)

- Non-compiling files will receive a 0 for all associated rubric items
- Do not submit `.class` files
- Test your code in addition to the basic checks on Gradescope
- Run Checkstyle on your code to avoid losing points
- Submit every file each time you resubmit
- Read the "Allowed Imports" and "Restricted Features" to avoid losing points
- Check on Piazza for a note containing all official clarifications